**A Case Study for the Database Design of**

**Event Management System**

Michael Arañas Sherwin Cabili

BSCS-III BSCS-III

September 8, 2013

**Abstract**

During the events in MSU-IIT the Event Management System will take place in checking the attendance of the students who are present in the event by their ID number. The students will just present their ID's to the administrator so that he/she can manually type ID number in to the system and to verify if that student is really the owner of that ID. Also, this system can be used by the Teachers/Lecturers/Professors to check the attendance of their student in their respective class. This system offers more convenience in contrast to the old way of sign-in/sign-out that consumes so much time.

**Contents**

**Abstract 1**

**1 Introduction**

* 1. Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  2. Description of the system . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  3. Scope and Limitation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1. **Requirements Analysis 5**

2.1 Data Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.2 Data Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.2.1 Entity-Relationship Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.2.2 Relational Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.3 Database Design and Normalization . . . . . . . . . . . . . . . . . . . . . . . . . . 13

**3 Database Testing**

3.1 Sample Interactive Queries . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.2 Application Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**4 Conclusion and Recommendation**

**A Code Listings**

1. **Introduction**
   1. **Background**

MSU-IIT still uses the old way of checking the attendance of the students during events and during classes. Writing their names in a piece of paper or using a MS-Excel software to list down their names, ID numbers, courses and other information which is very time consuming.

Due to this problem, we decided to design a database who will store all information needed specially the student’s ID number, name, year, course and etc. event’s name, date, sign-in/out time and student’s class schedules and to their respective teachers and create a simple web-based interface.

* 1. **Description of the System**

Event Management System is a web-based system that will be used by the student body counsel (the KASAMA and EC, different societies like ComSoc, Ecets, Jits, JIPIA, JIECEP, and etc.) or the Teachers/Lecturers/Professors to check the attendance of the students.

The teachers will create a Class which consists of Class Course number, Class schedule, semester, School year and Class room number. Teachers can only check attendance in the class not in the events. Teachers can only view class records not the event records.

The student body counsel or the Student Admin in the system will be the one that will create the event consist of Event name, date, semester, school year, venue, and the sign-in/out time. Student Admin will also checks the attendance of the students required to attend the event. They are not capable of checking the class attendance.

* 1. **Scope and Limitation**

Due to limited period of time, this study focuses only for the use in checking the attendance in MSU-IIT Events and classes. Also this study focuses only to the database design and sample queries

1. **Requirements Analysis Data Requirements**
   1. **Data Requirements**

**Student Account**

**Normal Student**

- In the event normal student will present their ID number to the administrator so that it can be manually type into the system.

- Is enrolled in the University so that the teacher can put him/her in the class he/she enrolled to.

**Student**

* Have records of his/her classes.
* Can view the said records of him/her.
* Exempted in attending events which he/she part of.
* Is predefined in the system
* Able to check the attendance of students in a given event given that he/she is part of the event.
* He/She can view the records of all the events he/she is part of.

**Teacher**

* Teacher will create a Class
* Class is another entity listed below
* Is maybe checking at least one class or none at all.
* If he is checking at least one class, he is able to view those records of the given class.
* Only the teacher can check a certain class could view its records and check the attendance of the class.

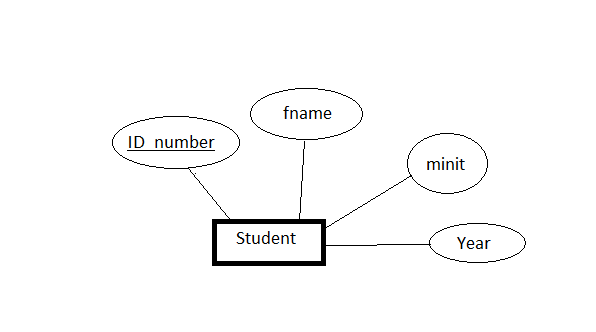
**Events**

* An event is created by a EC Officer assigned to the system
* Event has Event name, date, school year, sign in and sign out time and lastly the event venue.
* Event may or may not have sign-out.
* Events must be attended by the students who are required to attend it.

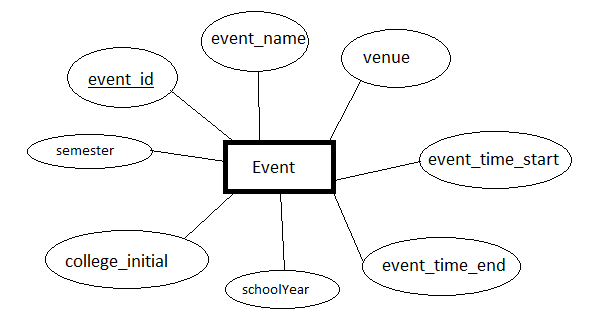
**Classes**

* Classes are created by the teacher handling the subject.
* Students in the Classes are manually populated by the teacher according to the list of students who have been enrolled in the University.
* Classes are attended by students enrolled in the said subject.
* Only teachers are allowed to check the attendance of a class.
  1. **Data Modeling**
     1. **Entity-Relationship Model**

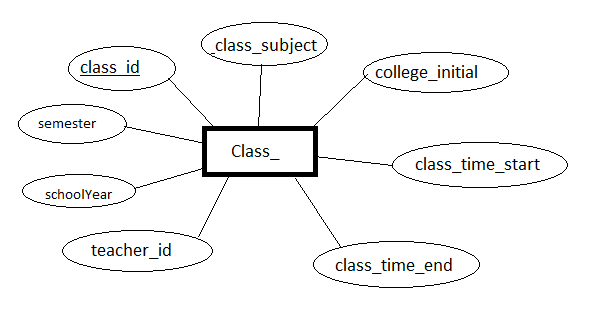
**Student Entity and its Attributes:**

****

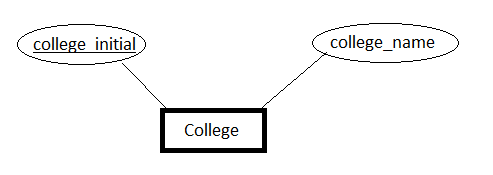
**Event entity and attributes:**

****

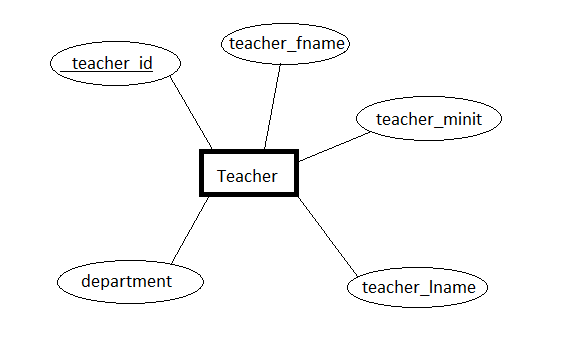
**Class\_ entity and Attributes:**

****

**College entity and attributes:**

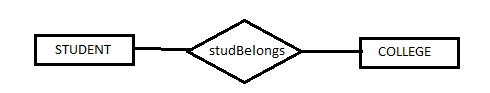
****

**Teacher entity and attributes:**

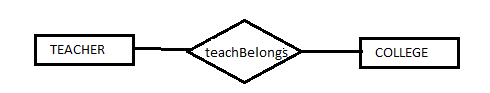
****

**2.2.1.2 Relationship Set Designation**

**Student belongs to College:**

****

**Teacher belongs to College:**

****

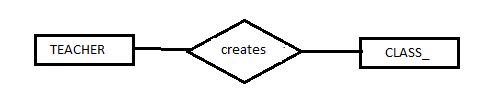
**Student attends Class:**

****

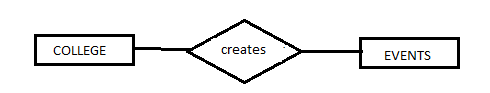
**Student attends Event:**

****

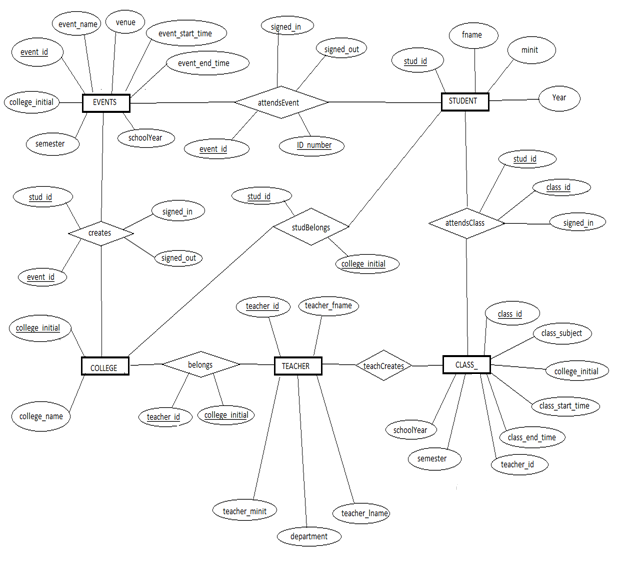
**Teacher creates Class:**

****

**College creates Events:**

****

**2.2.1.3 Entity-Relationship Diagram**

****

**2.2.2 Relational Model**

Student ( id\_number, fname, minit, lname, year);

Event ( event\_id, event\_name, venue, event\_time\_start, event\_time\_end, college\_initial )

Teacher ( teacher\_id, teacher\_fname, teacher\_minit, teacher\_lname, department)

Class\_ ( class\_id, class\_subject, college\_initial, class\_time\_start, class\_time\_end, teacher\_id)

College ( college\_initial, college\_name)

ScheduleDays (class\_id, days)

AttendsClass (stud\_id, class\_id, signed\_in, )

AttendsEvent( stud\_id, event\_id, signed\_in, signed\_out)

StudBelongs ( stud\_id, college\_initial)

TeachBelongs ( prof\_id, college\_initial)

**2.3 Database Design and Normalization**

**Student**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id\_number | fname | minit | lname | year |

**Teacher**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| teacher\_id | teacher\_fname | teacher\_minit | teacher\_lname | department |

**Event**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| event\_id | event\_name | Venue | event\_time\_start | college\_initial | semester | schoolYear |

**Class\_**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| class\_id | class\_subject | | | college\_initial | classRoom | class\_start | class\_time\_end | teacher\_id |
| semester | | schoolYear |

**AttendsClass**

|  |  |  |
| --- | --- | --- |
| stud\_id | class\_id | signed\_in |

**AttendsEvent**

|  |  |  |  |
| --- | --- | --- | --- |
| stud\_id | event\_id | signed\_in | signed\_out |

**College**

|  |  |
| --- | --- |
| college\_initial | college\_name |

**studBelongs**

|  |  |
| --- | --- |
| stud\_id | College\_initial |

**teachBelongs**

|  |  |
| --- | --- |
| prof\_id | college\_initial |

1. **Database Testing**
   1. **Sample Interactive Queries**

**Adding an Event in the Database:**

INSERT INTO event VALUES ('', '$event', '$room', '$time\_start', '$collegeInitial', '$sem', '$SY')

**Adding a Class in the Database:**

INSERT INTO class\_ VALUES ('', '$subject', '$collegeInitial', '$room', '$time\_start', '$time\_end', '$teacherID', '$sem', '$SY')

**Deleting an Event in the Database:**

DELETE FROM event WHERE event\_id = '$eID'

**Deleting a class\_ in the Database:**

DELETE FROM class\_ WHERE class\_id = '$cID'

**Display all Events:**

SELECT \* FROM event

**Display all class\_:**

SELECT \* FROM class\_

**Edit A class\_ information:**

UPDATE class\_ SET class\_subject = '$data' WHERE class\_id = '$cID'

UPDATE class\_ SET college\_initial = '$data' WHERE class\_id = '$cID'

UPDATE class\_ SET classRoom = '$data' WHERE class\_id = '$cID'

UPDATE class\_ SET class\_time\_start = '$data' WHERE class\_id = '$cID'

UPDATE class\_ SET class\_time\_end= '$data' WHERE class\_id = '$cID'

UPDATE class\_ SET teacher\_id = '$data' WHERE class\_id = '$cID'

UPDATE class\_ SET schoolYear = '$data' WHERE class\_id = '$cID'

UPDATE class\_ SET semester = '$data' WHERE class\_id = '$cID'

**Edit an event information:**

UPDATE event SET event\_name = '$data' WHERE event\_id = '$cID'

UPDATE event SET college\_initial = '$data' WHERE event\_id = '$cID'

UPDATE event SET venue = '$data' WHERE event\_id = '$cID'

UPDATE event SET event\_time\_start = '$data' WHERE event\_id = '$cID'

UPDATE event SET schoolYear = '$data' WHERE event\_id = '$cID'

UPDATE event SET semester = '$data' WHERE event\_id = '$cID'

* 1. **Application and Design**

1. **Conclusion and Recommendation**